View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NORA - Norwegian Open Research Archives

UNIVERSITY OF OSLO
Department of Informatics

Assisted curation of clinical trials using text mining tools

Master thesis

Michael Venables

November 8, 2013

Abstract

The amount of biomedical literature is growing rapidly, having nearly doubled the last decade. As the literature grows, it is becoming increasingly difficult for curators to keep up, as manual curation is a time-consuming process. Past research has indicated that computer-assisted curation can speed up this process considerably.

Our project aims to create a data mining approach using the ClinicalTrials.gov database as our source of data. Using external dictionaries and resources, new data will be added to the published trials by expanding on the pre-existing data tags and applying various text mining tools. As end product, we will develop a web-based curating tool that allow users to post more advanced queries for clinical trials than the currently existing interface at clinicaltrials.gov, as well as curate and add/edit manual annotations was developed. The system is available at http://invitro.titan.uio.no/clinicaltrials

Acknowledgments

I would like to thank my supervisors Sigve Nakken, Eivind Hovig and Geir Kjetil Sandve for their invaluable feedback and motivation through the writing of this thesis. Extra thanks to Sigve and Eivind for providing the scoring data used to evaluate the various text mining tools.

Chapter 1 Introduction

This chapter will cover the motivation and problem definition for the thesis, as well as providing relevant background information.

1.1 Background and motivation

The amount of biomedical literature is growing at an exponential pace; in the last decade, MEDLINE has grown at a ~4.7% annual rate. [1] As the literature grows, it is becoming increasingly difficult for curators to keep up as manual curation is a time-consuming process. Past research has indicated that computer-assisted curation can speed up this process considerably. [2]

Our project aims to create a data mining approach using the ClinicalTrials.gov database as our source of data. Using external dictionaries and resources, new data will be added to the published trials by expanding on the pre-existing data tags and applying text mining to the free text. As end product, we will develop a web-based curating tool that will allow users to post more advanced queries for published clinical trials than the currently existing interface at clinicaltrials.gov, as well as curate and add/edit manual annotations.

While there has been extensive focus on text mining MEDLINE articles and PubMed abstracts, there has been done far less text mining on clinical trials. There are new clinical trials published daily. This makes them an important source for making new discoveries about drug/gene/disease relationships. Additionally, trial temporary results are often accessible before the trial is officially completed, making available information that would normally not be published for months or years. [3]

This chapter will first provide an introduction to clinical trials, its history and characteristics, as well as the ClinicalTrials.gov database. Then, a summary of related work on text mining in biomedical literature will be given, followed by an introduction to literature curation and the purpose of curation tools.

1.2 Clinical trials

A clinical trial is a study conducted to evaluate the efficiency and safety of a new drug or treatment method. It is usually performed on a number of volunteers.

1.2.1 Brief history of Clinical Trials

One of the first clinical trials documented was performed in the eighteenth century. It was about the treatment of scurvy (a disease resulting from a lack of vitamin C) aboard a ship in the British navy. 12 patients were given 6 kinds of treatment. Two of the patients who were given citrus fruits (oranges and lemons) showed the most significant recovery.

The concept of randomization was first introduced in 1926. The first clinical trial that implemented it was done in 1931 by Amberson et al and experimented on a treatment for pulmonary tuberculosis. 24 patients were divided into two groups, where one was to receive the treatment of sanocrysin and the other was to receive merely distilled water as placebo. Randomization means the patients were distributed randomly in groups. This is important to avoid bias in the results. Blindness was also introduced in the same trial (see last section on blind studies). The patients in this trial were not aware if they were actually given the treatment of sanocrysin or in fact just distilled water, making it a single-blind study (see below). [4]

1.2.2 Characteristics of Clinical Trials

There are usually certain rules to decide who can and cannot sign up for a clinical trial, usually called eligibility criteria. Eligibility criteria often contains an age limit, a certain sex, or that the volunteer is diagnosed with a certain disease in a certain stage.

1.2.2 Characteristics of Clinical Trials

A clinical trial is normally conducted through several phases, usually three-four. In the first phase, side effects and results are considered, and if they're satisfactory, the clinical trial will proceed to the next phase. If the clinical trial completes the last phase and the results are satisfactory and the side effects are tolerable, it may replace currently used drug/treatments.

One should note that phases are not necessarily conducted consecutively or by the same institutions. Each phase uses a subset of tests that were tried and proven successful in earlier phases.

Some clinical trials include control groups. These are groups of participants who are given other treatment than the treatment in question, in order to provide something to measure the effectiveness of the new treatment against.

Control groups are usually divided using randomization, i.e. the patients are divided randomly in groups in order to avoid any bias. This can be as simple as flipping a coin or using (pseudo)randomly created numbers to delegate the patients into categories.

The control groups may receive no treatment at all, or the treatment that is currently being used. They are often treated with placebo, which is a treatment that resembles the treatment in observation but doesn't have the same effect (or any at all).

Usually, neither the patients or the investigators know which group are receiving placebo and which isn't in order to give more objective (unbiased) results. This is called a double-blind study. In single-blind studies, the patients are unaware whether they receive placebo or actual treatment while the investigators are informed. [4] [5]

1.2.3 The clinicaltrials.gov database

ClnicalTrials.gov contains over 150 000 trials from over 180 countries. Clinical trials from the database are primarily available as HTML documents. However, as data mining from the site is fairly common, it also conveniently provides the trials in raw XML formats. These records contain both structured information within uniform tags like «drugs» and «investigator», MeSH tags (Medical Subject Headings) as well as unstruc-

tured information like trial descriptions and other details. Each entry has information about the current phase, which countries are involved and eventually what interventions were done, e.g. what drugs were being tested. [5]

However, the information in the raw XML formatted trials are not necessarily unique to the one found in the HTML versions. While the latter formats have a full list of MeSH terms, including the parent nodes of the leaf nodes from the raw format, the XML versions include only the leaves. (This problem is covered in 3.2.4)

Another clinical trial database is the EU Clinical Trials Register [1] / EudraCT [2] database, where all EU countries are obliged to submit information about conducted clinical trials.

1.3 Related work on text mining in biomedical literature

In order to extract terms from the free speech text in clinical trials, both PoS tagging and dictionary-based approaches have been used, as well as simply using the terms mentioned in the trial description. Trials have also been manually annotated by clinical specialists in order to categorize them.

This has given results ranging from identification of previously unknown gene-drug-disease relationships to the development of tools improving the usability of or better representing the processed information. Here is a summary of various publications on text mining on clinical trials.

In 2011, Korkontzelos et al [6] [7] developed ASCOT, a tool providing a web interface for searching enriched trials. They employed part-of-speech tagging (see 2.1) to extract terms from free text parts. The main purpose of ASCOT is to simplify the process of creating eligibility criteria when creating new clinical trials. Typically, creating a clinical trial would include (for inexperienced users) searching for and processing previous trials in order to list the eligibility criteria. The tool aims to make this step less time-consuming.

1 https://www.clinicaltrialsregister.eu/
2 https://eudract.ema.europa.eu/

In 2008, Cao et al [8] published an article focusing primarily on clinical trials involving cancer vaccines. The aim of the project is to enable users to more easily process and understand the information in cancer related trials. As a result, they have developed a data mining approach that includes a search interface where users can find information about trials exceeding the basic web search at clinicaltrials.gov.

In 2012, Tasneem et al [9] published an article on the AACT, an enriched database containing additional meta data about the clinical trials. This project had two purposes; to improve the usability of the information in the clinicaltrials.gov trials and to find a method to classify studies according to clinical specialty. The trials were classified using the MeSH root nodes (see 3.2.4) as basis for the specialties.

Also in 2012, Lu et al [3] pointed out in their publication that a lot of the information found in trials are not published until several years after the trial has been conducted (avg. 5 years), if ever. This is expected as it takes time for trials to be conducted and published. Because of this, it is useful to find a way to systematically identify this information from the trials even before they are concluded. They hypothesized that clinicalTrials.gov is rich in information on "how genes affect drug responses in patients with specific diseases", or drug-gene relationships. A text-mining approach to identify relationships between drugs, genes and diseases was developed using ClinicalTrials.gov as their source.

1.4 Literature curation

Literature curation – or biocuration when applied on biomedical literature – involves translating and integrating information relevant to biology into a database or similar resource. The primary goals of biocuration are accurate representation of biomedical knowledge, as well as easy access to this data for researchers and scientists.

1.4.1 Manual curation

Manual curation implies that the curation process is done by a curator by reading the abstract / article / full text and registering curatable information. Manual curation of biomedical literature is both intensive and time-consuming. This coupled with the rate at which the literature is growing has created a need for literature curation tools(LCTs) – text mining based tools designed to accelerate the curation process.

1.4.2 Curation tools

Curation tools allow users to add terms to articles (e.g. a clinical trial or a PubMed article) in a more efficient way. In biomedical literature, these are typically genes, species, chemicals, diseases, etc.. The main purpose behind LCTs is to provide users with a faster way to annotate the articles than reading though every article and manually entering every annotation. Additionally, some LCTs processes the literature before making it available, extracting and registering annotations before a curator has even opened it. Most LCTs works by doing this automatic annotation through readily available text-mining tools. This way, the curators do not have to spend time registering the most obvious information. However, no LCTs so far have achieved the precision as expert curators, and there are sporadic false positives. This introduces the curator with a new task; verifying (or invalidating) the automatically registered annotations. Additionally, curators have to make sure there is no missing information that the algorithms have missed. Still, this is in most cases less time-consuming than manually entering every annotation. Most LCTs include a tool that enables the users to validate or invalidate these (correct for false positives, a result of lacking precision). Eventually, users can look for annotations that have been overlooked by the algorithms (a result of lacking recall) and use the system to register them.

PubTator, a LCT developed at the National Center for Biotechnology Information, is one of the highest rated (highest reported F-measure, highest reduction in curating time) biomedical literature curation tools. It provides several types of annotation of PubMed articles; document triage, entity annotation and relationship annotation. Users can look up abstracts by terms as well, i.e. genes, chemicals and diseases.

1.4.2 Curation tools

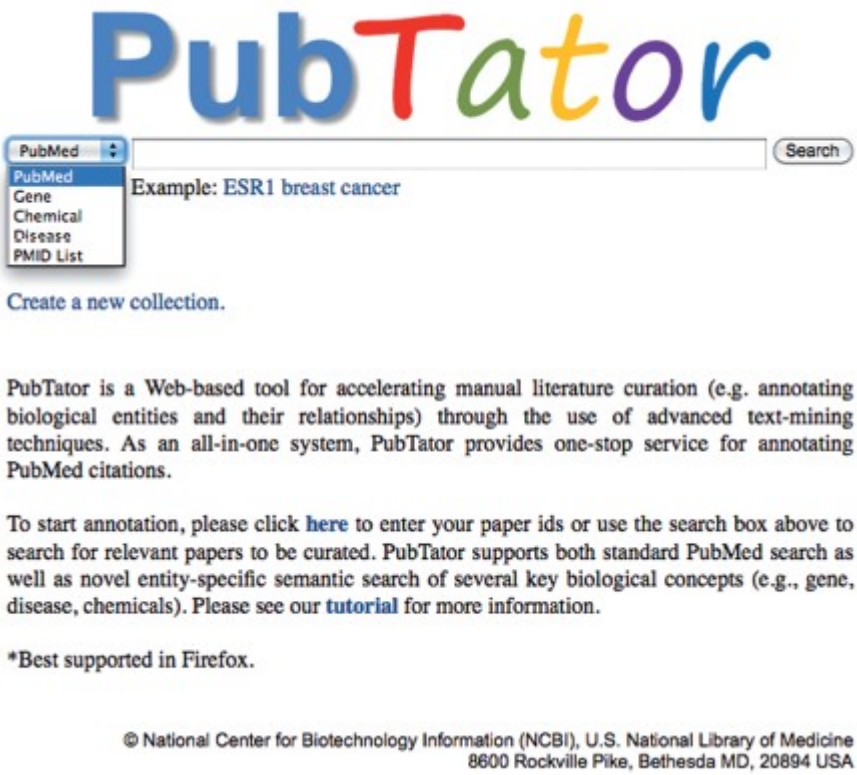


Illustration 1.1: It features an interface very similar to PubMed in order for users, already familiar with the PubMed interface, to avoid having to familiarize themselves with a new system.

PubTator keeps the entire content of PubMed and annotates new articles daily. It employs various text-mining tools to automatically annotate genes, species, mutations, chemicals and diseases. In document triage, users can prioritize which articles are deemed curatable. Users can then curate the articles and validate/invalidate these annotations, as well as adding their own. As text-mining tools are not 100% correct, there is a possibility of false positives (in addition to a lack of recall) in the annotations. As such, it is useful for users to manually remove certain annotations. For instance, as shown in Fig. 1.7, in PubMed article PMID 23702042, the abbreviation for the chemical Praeruptorin D (PD) has mistakenly been annotated as Parkinson's Disease from this sentence: “Praeruptorin D (PD) is one of the major active constituents of Peucedanum praeruptorum Dunn (Qianhu)”.

1.4.2 Curation tools

Entity type	Entity mention	Concept ID	Nomenclature	Delete
Gene	Cyp3a11	13112	NCBI Gene	Delete
Gene	CYP3A4	1576	NCBI Gene	Delete
Gene	CYP3A	1574	NCBI Gene	Delete
Species	human	9606	NCBI Taxonomy	Delete
Species	mice	10090	NCBI Taxonomy	Delete
Gene	orphan nuclear receptor	2103	NCBI Gene	Delete
Disease	PD	168600	MEDIC	Delete
Gene	Pregnane X receptor PXR	8856	NCBI Gene	Delete

Illustration 1.2: The PD (Parkinson's Disease) annotation is actually a false positive.

We can confirm that PD is annotated here for Parkinson's Disease by looking up the given ID in the MEDIC database. [10] Using the PubTator interface, this can easily be fixed simply by deleting the annotation and (possibly) adding the correct chemical annotation as shown in Fig. 1.8.

Entity type	Entity mention	Concept ID	Nomenclature	Delete
Gene	Cyp3a11	13112	NCBI Gene	Delete
Gene	CYP3A4	1576	NCBI Gene	Delete
Gene	CYP3A	1574	NCBI Gene	Delete
Species	human	9606	NCBI Taxonomy	Delete
Species	mice	10090	NCBI Taxonomy	Delete
Gene	orphan nuclear receptor	2103	NCBI Gene	Delete
Gene	Pregnane X receptor PXR	8856	NCBI Gene	Delete
Chemical	Praeruptorin D		MESH	Delete

Illustration 1.3: False positive annotation fixed

PubTator is acknowledged as one of the best LCTs, and was rated highest and was most recommended among all the participants in the interactive text mining track of BioCreative 2012 workshop. [11] It is proven to accelerate the curating process as well as increase the accuracy at which annotation is done.

1.4.2 Curation tools

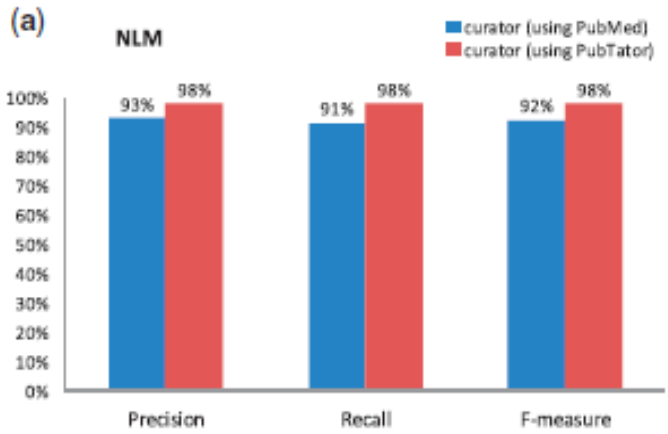


Illustration 1.4: PubTator annotation performance vs manual annotation.

Both precision and recall was increased using PubTator, raising the F-measure (the harmonic mean of precision and recall – see 2.2.3 Evaluating a NER system for details) to 98% compared to the 92% when annotating manually. Curating time also decreased from an average of 326s to 190s, a 42% increase in efficiency.

MyMiner [12] is another literature curation tool. It provides a simple web interface combining PHP, JavaScript and AJAX. It includes triage tasking, classification and annotation. The system automatically annotates entities through text-mining tools, specifically the ABNER[13] tagger for proteins, DNA, RNA, cell lines and cell types and the LINNAEUS[14] system for species and organisms. Additionally, the text-mining systems can be modified to detect user-defined entities if a dictionaries are provided. Automatically annotated entities can be edited by users and false entities can be removed completely. It also provides an interface for implying relationships between entities. In addition, MyMiner provides the possibility of generating training data for document categorization. MyMiner was evaluated to manual text annotations generated by unassisted or assisted human annotators, showing a decrease in annotation time (up to 90%, averaging on 70%), with no difference in the quality of annotations.

The NCBO Annotator[15] is another web-based annotator. It is developed by National Center for Biomedical Ontology in collaboration with the National Center for Integrative Biomedical Informatics.

1.4.2 Curation tools

Several other literature curation tools have been developed that are not meant for biomedical text, but rather general text, such as brat (brat rapid annotation tool).

Chapter 2 Text Mining

While some of the information in the trials is easily available, mutations and genes are not found in specific locations of the trials, but rather in free text. Extracting these are not always as straightforward as extracting already indexed information like drugs and MeSH terms. To approach this, we will use various text mining techniques.

2.1 POS tagging

When processing natural language, one of the most fundamental stages is marking up each word into categorizations such as verbs, nouns, adjectives, etc. Additionally, several words can make up chunks, or phrases. For example, a noun following an adjective would make up a noun phrase. A verb leading a a noun or a noun phrase would constitute a verb phase. As a rule (in English), a sentence has to contain a noun phrase followed by a verb phase to be valid. The process of categorizing tags into chunks is called chunking.

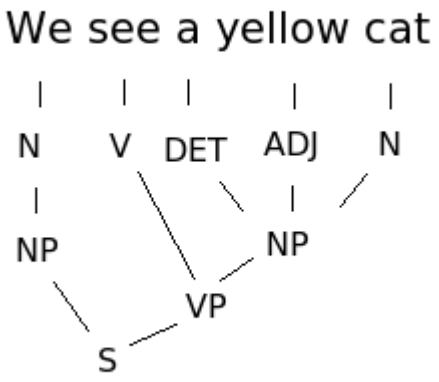


Illustration 2.1: A simple example of a POS tagged sentence

While most words only have one meaning and in in most cases only one category, many words also have two or more different meanings. As such, certain words are ambiguous, and it is not as straightforward as simply looking at the word itself and assigning a category; the context (sentence) has to be considered as well, increasing the process' intensity several times. Consider the sentence “they refuse to permit us the refuse permit”. Here “refuse” appears both as a verb and as a part of the noun phrase “the refuse permit”, depending on the context.

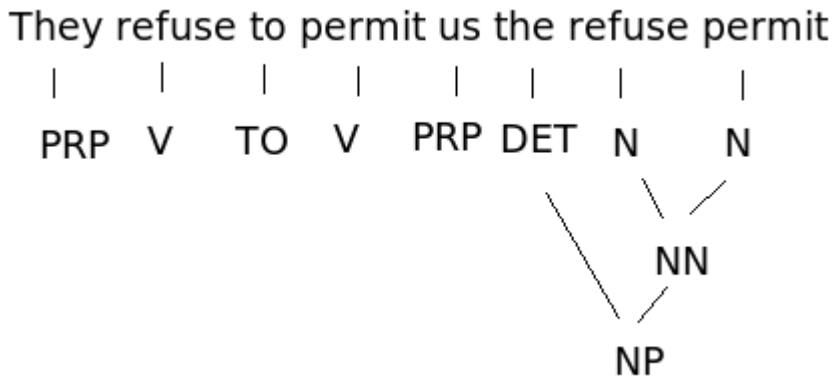


Illustration 2.2: "Refuse" and "permit" appearing with different tags, depending on the context. (complete chunking omitted for simplicity)

Even sentences containing only unambiguous words can be ambiguous as a whole and represent different meanings. Consider the sentence “I shot an elephant in pajamas.” Intuitively we assume that “I” was wearing pajamas and shot an elephant. Alternatively, though, the sentence could also imply that the elephant was wearing pajamas. [16]

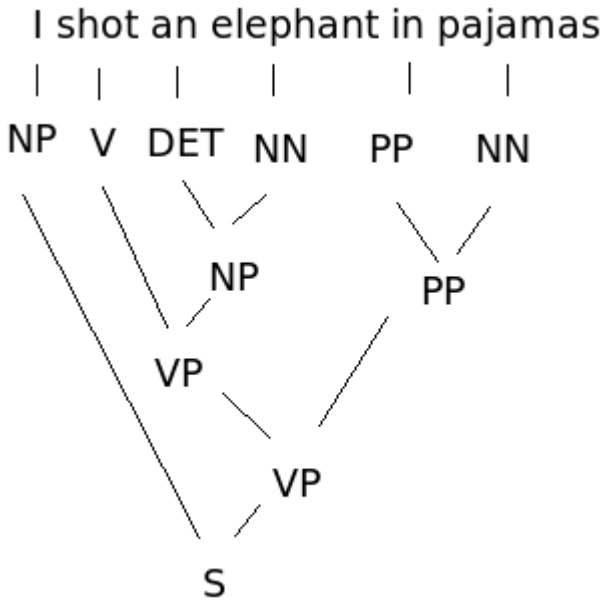


Illustration 2.3: "I" shot an elephant while wearing pajamas

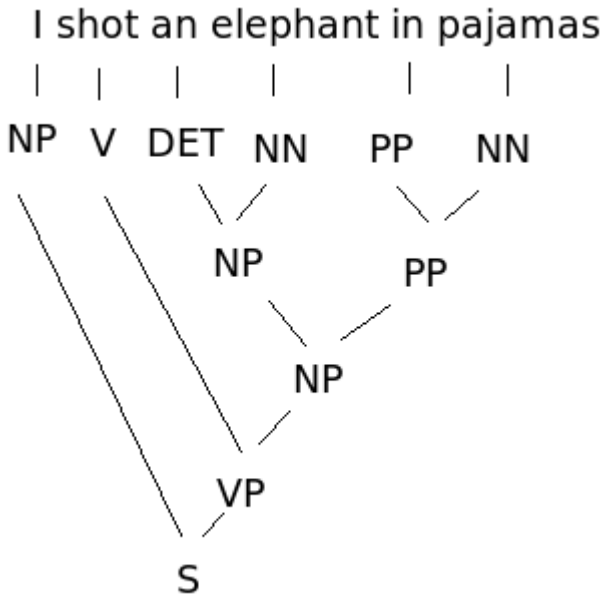


Illustration 2.4: "I" shot a pajamas-wearing elephant....

Even though the tags are identical, the chunks are different. This is caused by the ambiguous bindings (or “prepositional phrase attachment ambiguity”); the noun phrase can bind with the preceding verb to form a verb phase, or it can bind with the following prepositional phrase to form another noun phrase. One approach is to evaluate every possible permutation of phrases and find out which ones constitute a sentence (containing a noun phrase and a verb phrase), but this is very time-consuming and not feasible when dealing with the amount of text we are. As an alternative, it is possible to train programs to do this for us, and there exists several kinds of POS taggers.

2.1.1 Hidden Markov Models

Markov Models are made by and named after Andrey Markov. They work under the assumption that “the future is independent of the past, given the present.” In practice this means that if we know the exact state of the present, we can calculate the most probable state of the future regardless of the past states, i.e. knowing the past states will not give us any more information, as the information is encoded in the present state. Markov Models are used for predicting the most probable future outcome in many fields, including weather, finance, music, language, etc. We will take a look at how this (or more specifically *Hidden* Markov Models(HMMs)) can be applied specifically in natural language processing / text mining to drastically decrease the complexity of POS tagging after we look at some general examples.

> “For the most reliable weather forecast for tomorrow, look out your window.”

Hidden Markov Models, in contrast to regular Markov Models, contain hidden events for each observable step (e.g. a POS tag for each word in a sentence). A very simple example of HMM application is the weather example. Here we use a HMM to predict the weather each day (hidden events), given only the amount of ice creams that was eaten each day (observable states).

2.1.1 Hidden Markov Models

The complete structure of a HMM is outlined here:

- $Q = q_1, q_2, ..., 1n$; a set of N hidden states
- $A = a_{11}, a_{12}, ..., a_{n1}, a_{nn}$: a transitional matrix with the probabilities of moving from state *i* to state *j*
- $O = o_1, o_2, ... o_T$: a sequence of *T* observations
- $B = b_i(o_t)$: emission probabilities, meaning the probabilities of the observation o_t given the current state *i*
- q_0, q_F: start and final state that are not associated with observations

The first step of applying a HMM is to train it on a given training set, i.e. the sequence of observable states and the corresponding hidden events. In this example these are represented by the observed ice cream consumption for each day along with the weather.

2	H
2	H
2	H
2	H
3	C
2	C
1	H
3	H
3	H

This is an excerpt from what a typical data set would look like. We have here a sequence of four consecutive hot days where 2 ice creams were consumed, followed by a cold day where 3 ice creams were consumed. This is actually counter-intuitive and “noisy”, as hot days will on average have higher ice cream consumption. The second sequence is more representative of the whole data set as the hot days there mostly have

higher ice cream consumption. Looking at the weather in these sequences, we can calculate the overall transitional probabilities, i.e. the probability of the weather changing from hot to cold or cold to hot and the probability of a sequence ending or starting with a cold or hot day. Additionally, by looking at the ice cream consumption, we can calculate the emission probabilities, i.e. the probability of hot or cold weather given the amount of ice creams eaten. We process the data set and calculate the emission and transitional probabilities using the following formulas:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

The transition probabilities are assigned simply by looking at the ratio between the number of times a certain transition from state t_{i-1} to t_i has occurred in the training set and the total number of occurrences of state $t_{i-1.}$ Similarly, the emission probabilities are calculated by looking up the ratio between the number of times the hidden event w_i has occurred along with the state t_i and the total number of occurrences of state $t_{i.}$ Some examples of data after we have processed the complete data set:

- The probability of starting a sequence of days on a hot day is 76%
- The probability of a hot day following a cold day is 26%
- The probability of cold weather when 3 ice creams are consumed is 14.1%
- The probability of cold weather when 2 ice creams are consumed is 34.1%
- The probability of cold weather when 1 ice cream is consumed is 50.4%

2.1.1 Hidden Markov Models

By looking at this information we can intuitively make several assumptions; most days in the data set are hot, a day with a given weather is likely followed by a day with the same weather, and that the probability of cold weather decreases as the ice cream consumption increases.

We now have a fully trained HMM for predicting the most likely corresponding weather for a sequence of days given the amount of ice cream consumption per day. We will have a look at how to apply this HMM to unannotated sequences. First we need a basic understanding of Bayes Rule:

$$P(A, B) = P(A)\, P(B \mid A)$$

inversely, we also get:

$$P(A, B) = P(B)\, P(A \mid B)$$

therefore:

$$P(A)\, P(B \mid A) = P(B)\, P(A \mid B)$$

->

$$\underline{P(B \mid A) = P(B)\, P(A \mid B) / P(A)}$$

Bayes Rule lets us compute P(B|A) in terms of P(A|B), and makes it possible to make the formula for applying the HMM more easily computable. Let's take it from the beginning: the task of the trained HMM is to take an observed sequence of steps (in this case, number of consumed ice creams) and determine the most likely corresponding sequence of hidden states (weather).

$$\hat{t}_1^n = \arg\max_{t_1^n} P\left(t_1^n | w_1^n\right)$$

We then apply Bayes rule to turn the formula into something more tractable;

2.1.1 Hidden Markov Models

$$\hat{t}_1^n = \arg\max_{t_1^n} P\left(t_1^n | w_1^n\right) = \arg\max_{t_1^n} \frac{P\left(w_1^n | t_1^n\right) P\left(t_1^n\right)}{P\left(w_1^n\right)}$$

Instead of computing the probability of a hidden state given an observation, we compute the probability of an observation given a hidden state. Since the denominator is equal for every word, we can leave it out of the equation. We now have:

$$\hat{t}_1^n = \arg\max_{t_1^n} P\left(t_1^n | w_1^n\right) = \arg\max_{t_1^n} P\left(w_1^n | t_1^n\right) P\left(t_1^n\right)$$

Now we can start estimating the terms. We make two underlying assumptions;

- The probability of a word is only dependent on its part-of-speech tag:

$$P(w_1^n | t_1^n) \approx \prod_i^n P(w_i | t_i)$$

- The probability of a tag is only dependent on the immediately preceding tag (the Markov assumption):

$$P(t_1^n) \approx \prod_i^n P(t_i | t_{i-1})$$

This leaves us with the final formula:

$$\hat{t}_1^n \approx \arg\max_{t_1^n} \prod_i^n P(w_i | t_i) P(t_i | t_{i-1})$$

Now, this can be directly applied either to determine the probability that a given sequence of observations will produce a given sequence of hidden states, or it could be used to find the most likely sequence of hidden states produced by a given sequence of observations. [17] There exists several approaches for this, but it will not be covered here.

2.1.2 Application in biomedical text mining

While the concept of POS tagging and chunking is not directly related to bioinformatics, it can in theory be implemented when looking for entities in free text such as genes and mutations using a dictionary-based approach. Since genes and mutations appear as nouns or noun phrases, we can use POS tagging / chunking to mark parts of the text that should be looked up. In comparison to looking up every possible subset of the text, this will be much more efficient. [18]

This assumes that the chunking of a sentence plus looking up the resulting marked nouns and noun phrases is not more time-consuming than simply looking up every subset manually. We can make this assumption by comparing the number of look ups that are required per string by each operation. Using the NCBI dictionary for human genes / proteins [19], there are currently over 46 000 entries. Each of these contain a symbol, a description as well as unofficial symbols and alternate descriptions. On average, there are 4.89 symbols and/or descriptions for each entry. This means that for each look up operation, the string is compared to roughly 220 000 symbols/descriptions. Since we are matching every possible subset, we have to perform $(n^2/2)+n$ number of look ups per string, where n is the number of tokens in the string. When chunking, we perform the same number of look ups(using CKY algorithm), yet here we only have the grammar rules to consider, which are much more limited than the number of symbols/descriptions. For comparison, the Penn Treebank [20], which has tagged 7 million words of text, contains only 65 unique POS tags. Example:

2.1.2 Application in biomedical text mining

" This phase I part of the trial is justified by a possible interaction of the two drugs that are substrates of cytochrome P450 CYP3A4 "

This sentence contains a mention of the protein(enzyme) CYP3A4. Processing this sentence manually would require 312 look ups vs just looking up the nouns and noun phrases after POS tagging the sentence.

2.1.3 Challenges when POS tagging clinical trials

When processing the free text in clinical trials, the section covering the inclusion / exclusion criteria for participation may contain mentions of genes and mutations that are not necessarily represented in the brief summary or detailed description of the trial. Therefore, the eligibility criteria section should also be processed. However, these sections are typically structured like bullet points for each criteria. Here's an example excerpt from the criteria section of a trial [3]:

Inclusion Criteria:

- Age 6 to 75 years old
- Established diagnosis of CF with at least one abnormal G551D-CFTR allele
- Eligibility for and intent to start treatment with ivacaftor or started treatment with ivacaftor within previous 6 months

POS chunking is done on each sentence independently. These bullet points will be processed as sentences, yet they are not structured like conventional sentences, but rather as items in a list. This is not optimal as general POS taggers will not be trained on this format. However, as the full parse tree is not necessarily required, a bottom-up approach would be able to fill out the lower part of the parse tree. While this tree would never be complete, the bottom-level noun phrases would still be found.

3 http://clinicaltrials.gov/ct2/show/NCT01549314

2.2 Named entity recognition

Named entity recognition (NER) is a common approach for automatically annotating entities in free text. It combines the problem of recognizing entities in text, assigning them to a class (classification) and reducing synonyms to a preferred name (normalization). For annotating genes and mutations, we will look at various approaches, including several third-party NER systems.

Typical NER systems apply rule-based[4], dictionary-based[5] and machine learning based[6] methods. Some systems include natural language processing, such as POS tagging / chunking (see 2.1). Dictionary-based approaches simply use a set of names to identify entities in the text. These names can then simply be matched exactly against the text, which is very precise yet yields a very low recall. Other possibilities are generating variants of spelling to compensate for typographical errors in the text. The advantage of dictionary-based approaches is the lack of need of a pre-annotated corpora. The downside is that context is not taken into consideration. Rule-based approaches use rules to separate different classes. Rule-based approaches suffer from a strong trade-off between precision and recall. Applying highly specific rules will result in high precision, but a very low recall as most entities will not be detected. Conversely, using very general rules will detect most entities but also many false positives, i.e. low precision. Machine learning based approaches use a training corpus to perform statistical analysis to deduce the most likely context in which an entity would appear. They often implement POS tagging / chunking. The downside of machine learning based approaches is the training time and the dependency of annotated corpora. [21]

2.2.1 Machine learning

4 e.g.: MutationFinder
5 Basic symbol matching (see 3.2.6)
6 e.g.: BeCAS, Gimli

2.2.1 Machine learning

To understand how a machine learning based named entity recognizer is developed, a basic introduction to the field is useful. Machine learning is the method of “teaching” an algorithm to perform tasks by looking at example sets of data. This way, the algorithm does not have to be explicitly programmed to solve the problems.

Machine learning is generally divided into two categories; supervised and unsupervised. In supervised learning, the training data is labeled, and the desired output is known. The algorithm will then calculate estimates by processing these data and approximate a function to determine unseen input in unlabeled data sets. An example of supervised learning is the k-NN (k nearest neighbor) algorithm. This algorithm classifies data by looking at the class of the k points which have most similar attributes to the target. The learning aspect in k-NN consists of creating classes with attributes that can be directly compared with the unlabeled data points in order to measure similarity.

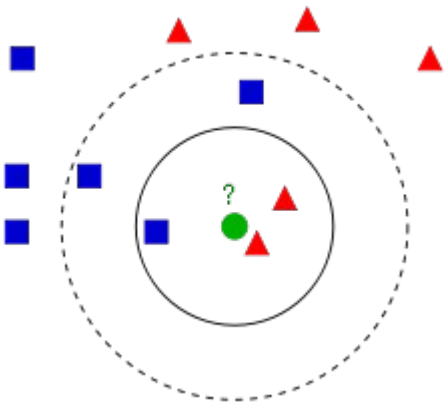


Illustration 2.5: A simple k-NN example

Fig. 2.5 illustrates a k-NN classification. If k is set to 3, the target point will be classified as a triangle (2 triangle vs 1 square neighbor). If k is set to 5, it will be classified as a square (3 squares vs 2 triangle neighbors).

In unsupervised learning, no desired output is pre-defined, and the algorithm attempts to discover structure in the unlabeled data. An example of unsupervised machine learning is clustering. As opposed to classification(e.g. K-nn), clustering attempts to create classes among data points instead of assigning the data points to pre-defined classes.

2.2.1 Machine learning

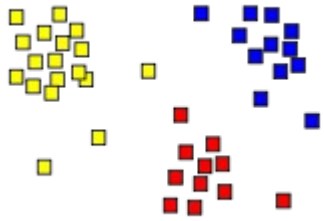


Illustration 2.6: A simple clustering example

Typically, the training data is divided into a training set and a test set / validation set in order to measure the algorithm's performance. This is easily measured as the test set, as a part of the training data, is already labeled. As such, no extra effort is required to rate the performance of the algorithm.

When training a supervised learning algorithm, training data should be sufficiently big in order to minimize potential noise in the data. Additionally, the algorithm should not perform the learning phase for too long, as it may adjust to very specific noisy features of the training data. This concept is called overfitting, implying it is overly fitted specifically to the training data. A clear sign of overfitting is when the algorithm's performance on the training data keeps increasing but decreases for unseen data. This point is the optimal time to end the training process.

Cross-validation is one way to deal with overfitting. This means dividing the training data into folds of training sets and test sets (typically one fold assigned as test data and the rest as training data).

A concrete example of a supervised learning task is the classification of electromyographic (EMG) signals corresponding to various hand motions done in a study by K. Glette et al in 2008 [22] demonstrating the potential of evolvable hardware as prosthetic controllers. The data is divided into three data-sets (one each day) and contains 40 different signals per motion (10 readings for each of 4 sensors), as well as one classification value (label) indicating which of 8 hand motions is being made.

2.2.1 Machine learning

We will write a trainable program and apply the k-NN method on the data sets, demonstrating the use of cross-validation. First the algorithm processes the training data, each day divided into separate files formatted with one hand motion for each line. Each line includes the label following 40 separate readings. Both label and readings are represented with numbers. Specifically, each of the 8 hand motions are represented by the numbers 1-8, while readings are represented as decimals ranging from 0-1. First we have to find an approach for approximating how similar readings are to each other. Since each reading consists of a matrix of values, they can be treated as points in a 40 dimensional coordinate system. Consequently, we can approximate the similarity by calculating the distance between them. For this, we will use the Euclidean distance, which is basically the Pythagorean theorem extended to *n* dimensions:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_i - q_i)^2 + ... + (p_n - q_n)^2}.$$

The similarity of two points then can then be calculated from the distance (e.g. d^{-1}, 1-d, etc). The training process of the algorithm consists of loading the training data in order to later use the “coordinates” as neighbors when classifying unlabeled data. When given unlabeled input, the k-NN algorithm assigns the point to the class which is most prominent among its *k* neighbors, i.e. the *k* most similar points from the training data.

To evaluate our k-NN algorithm, our training data is divided into a training set and a validation set. In the simplest case, this is done simply by splitting the original set arbitrarily. However, the evaluation results obtained this way tends to be affected by how the data is split. To reduce this bias, we will use cross-validation. This means using different “folds” as test set. In our case, we will divide the training set into 5 folds, and use each fold as a test set once. Results:

Data set	**Accuracy**	**Standard dev.**
Day 1	95.8%	8.4%
Day 2	92.0%	11.2%

Day 3	96.0%	9.7%
Day 1-3	93.9%	7.2%

Here we are using the same sets for training and evaluation (including cross-validation). The table shows the accuracy of the algorithm as well as the standard deviation of the results from the cross-validation. In this problem, the k-NN is able to score relatively high. Typically, less sophisticated algorithms such as k-NN will have a lower score on more complicated training data.

2.2.2 Training a named entity recognizer

Being a machine learning based approach, NER systems are trained by processing a pre-annotated corpus, or training set. As an example, let's look at the methods for training a simple NER with a few classes; names, locations and organizations. Creating a training set for this NER would mean tagging a huge amount of free text with occurrences of these classes. How these occurrences are tagged are not important, as long as the learning algorithm knows how to make them out, i.e. they should not be ambiguous or contain characters that can be mistaken as part of the text itself. Let's look at an excerpt from a typical training set. Original text:

"The fate of Lehman Brothers, the beleaguered investment bank, hung in the balance on Sunday as Federal Reserve officials and the leaders of major financial institutions continued to gather in emergency meetings trying to complete a plan to rescue the stricken bank. Several possible plans emerged from the talks, held at the Federal Reserve Bank of New York and led by Timothy R. Geithner, the president of the New York Fed, and Treasury Secretary Henry M. Paulson Jr. " [7]

And then after the original text has been annotated for the purpose of training our simple three-class NER:

7 Example from the Stanford NER (http://nlp.stanford.edu/software/CRF-NER.shtml)

2.2.2 Training a named entity recognizer

“The/O fate/O of/O Lehman/ORGANIZATION Brothers/ORGANIZATION ,/O the/O beleaguered/O investment/O bank/O ,/O hung/O in/O the/O balance/O on/O Sunday/O as/O Federal/ORGANIZATION Reserve/ORGANIZATION officials/O and/O the/O leaders/O of/O major/O financial/O institutions/O continued/O to/O gather/O in/O emergency/O meetings/O trying/O to/O complete/O a/O plan/O to/O rescue/O the/O stricken/O bank/O ./O

Several/O possible/O plans/O emerged/O from/O the/O talks/O ,/O held/O at/O the/O Federal/ORGANIZATION Reserve/ORGANIZATION Bank/ORGANIZATION of/ORGANIZATION New/ORGANIZATION York/ORGANIZATION and/O led/O by/O Timothy/PERSON R./PERSON Geithner/PERSON ,/O the/O president/O of/O the/O New/ORGANIZATION York/ORGANIZATION Fed/ORGANIZATION ,/O and/O Treasury/ORGANIZATION Secretary/O Henry/PERSON M./PERSON Paulson/PERSON Jr./PERSON ./O ”

The format of the training set is straightforward and intuitive; each word, or token, is followed by a slash and then either a given class or a 0 if no class is assigned. Another typical tagging system is to encapsulate every entity in XML tags, e.g. “<PER>Henry M. Paulson Jr.</PER>”. This eliminates the need for a tag for every class-less word and makes it more clear where an entity begins and ends.

Given this information, the NER calculates estimations that will determine what words are tagged when it is later run on un-annotated text. When learning from the training set, our tagger will consider the position of the class, i.e. preceding and succeeding words/classes (N-grams), and of course the word itself. More advanced NER systems will use several other features such as POS tagging, conjunctions(disambiguation of candidates containing “and/or”) and many more.

Naturally, annotating such corpora of sufficient size is both a time-consuming and expensive effort, making them scarce and valuable. Research has been made on the possibility of combining different corpora with only partially semantic overlap as a single training set to make up for this. [23]

2.2.2 Training a named entity recognizer

2.2.3 Evaluating a NER system

To measure the performance of a named entity recognition system, the traditional F-score (or F-measure) system is usually used. This score evaluates the recall and precision of the system. The recall is the number of correct results divided by the number of correct results that should have been found, while precision is the number of correct results divided on the number of total results. In short, recall reflects how many of the results the system manages to pick up, while precision is basically a measurement of noise in the results.

The F-score itself ranges from 0 to 1 (often translated into percent) and is calculated by as the harmonic mean of precision and recall. The harmonic mean of two numbers is calculated simply as the product of the numbers divided by the sum of the numbers, multiplied by two.

$$H = \frac{2x_1x_2}{x_1 + x_2}.$$

Some examples of evaluation of NER systems:

System	**Precision**	**Recall**	**F-Measure**
Gimli NER	90.22%	84.32%	87.17%
BANNER	85.09%	79.06%	81.96%
ABNER	83.21%	73.94%	78.30%
LingPipe	60.34%	70.32%	64.95%

The F-score is not used only on NER systems, but is applied in general in statistics to measure accuracy.

2.2.4 NER in the biomedical domain

2.2.4 NER in the biomedical domain

The recognition of biological concepts in free text is problematic due to several factors. There are no naming conventions for gene / protein names, which makes it difficult to establish general rules when looking for them. Additionally, there are typically several different names used for each gene / protein, including alternate descriptions which may consist of up to several words. In many of these descriptions, it is not always clear where the gene / protein name starts and ends, such as 'cellular retinoic acid binding protein 1'. According to the BioCreative corpus of expert-tagged gene names, 53% of all names consists of more than one token, so this problem is not unusual. Another prominent problem when looking for genes/proteins are names identical to common acronyms or words. MRI, which is an alternative symbol for the gene C7orf49, is identical to the acronym for magnetic resonance imaging. CSF, one of the alternative symbols for both CSF2 and LAMC2 gene, is identical to the acronym for cerebral spinal fluid. Additionally, there are gene symbols which are outright idiotic, such as STOP, IF, FISH. [21]

While many NER packages are designed to classify several entities (genes/proteins, mutations diseases or drugs), most packages are class specific, meaning they are designed only to find one particular type of entity. Unlike gene / protein names, certain types of mutation names are named more systematically and can more easily be retrieved using simple regular expressions. Specifically, point mutations are named in the following format: a letter indicating the amino acid that is replaced by the mutation, followed by a number indicating the position of the mutation, followed by a letter indicating the amino acid that has replaced the original. For example, the point mutation G551D would indicate “a glycine[G]-to-aspartic acid[D] missense mutation at codon 551”.

There has been significant progress recent years in biomedical named entity recognition(BNER). Several freely available BNER systems are available, such as BANNER[24], ABNER[13], LingPipe and of course BeCAS and Gimli which have been evaluated on clinical trials in this thesis and will be covered at the end of the chapter.

2.2.5 Challenges specific to clinical trials

As with POS tagging, the bullet point format of the criteria section is not ideal for NER packages as they are trained on typical sentences. This may cause a lowered precision and recall.

2.2.6 GNAT

One of the NER packages we will evaluate on clinical trials is GNAT[8], an open source Java library developed by Hakenberg et al. In addition to named entity recognition, the package provides normalization of gene and protein mentions. GNAT features a web service that can be used directly through a browser by providing arguments in the URL. Example:

http://bergman.smith.man.ac.uk:8081/?text=human%20p53%20protein&task=gnorm

In this example, the task would be normalization of the string “human p53 protein”. However, this web service doesn't really seem to be a fan of running, which obviously will cause problems. As of November 2013, the web interface seems to have been moved to http://cbioc.eas.asu.edu/gnat/start.html, where posting a query still returns nothing but an error message. GNAT also provides a programmatic / command-line approach. Unfortunately, this approach also seems to be dependent on the same server as the web service. [25]

2.2.7 Gimli

8 http://gnat.sourceforge.net

Gimli[9] is another open source Java library for named entity recognition. It is developed by David Campos et al. This package is highly customizable and includes the possibility of training customized models. These models can then be combined or run individually when processing text. Gimli is also dependent on a remote web service, but fortunately it tends to stay running. [26]

2.2.8 BeCAS

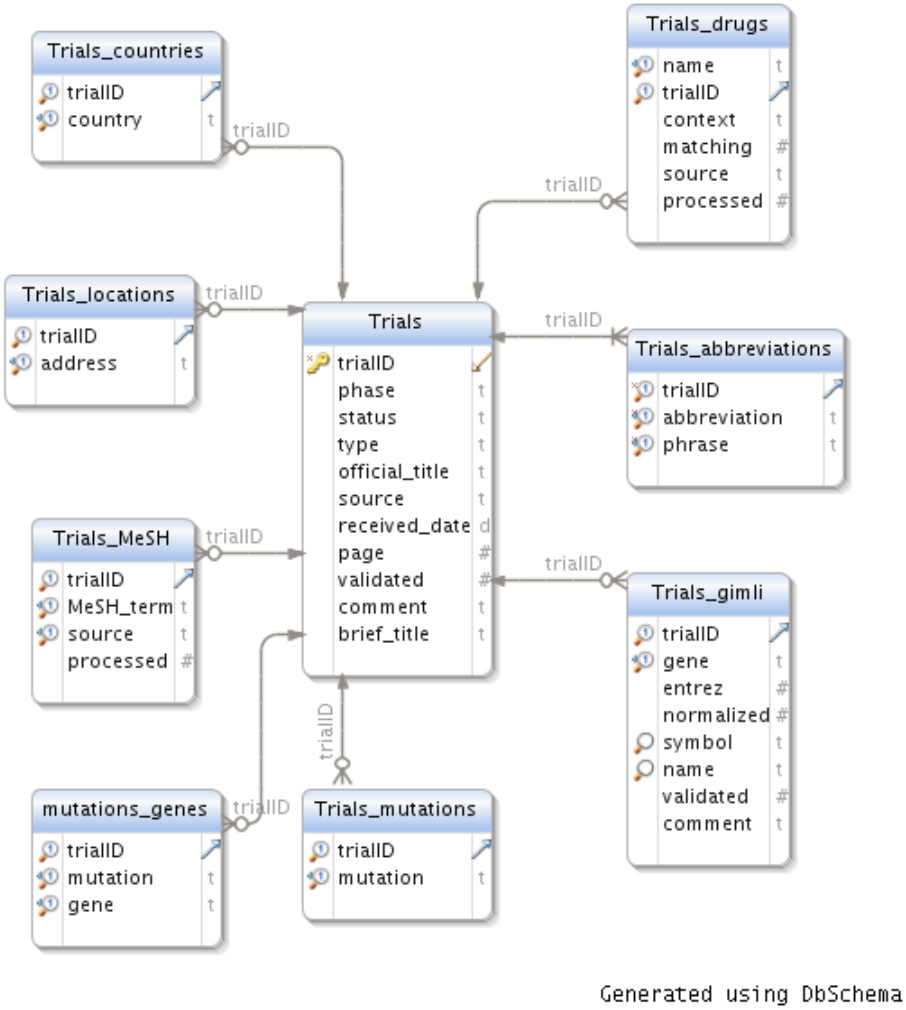
BeCAS is another web-based service for recognizing biomedical concepts. It is developed by a team consisting largely of the same developers / researchers as the Gimli team. However, while Gimli seems to have a focus on customizable training models, BeCAS is more focused on addressing multiple concept types and using external databases. [27]

9 https://github.com/bioinformatics-ua/gimli

Chapter 3 Enriched Trial Database

3.1 Design

To store the information about the trials, a back-end MySQL database was created. The database is divided into separate tables with a main table containing the basic information and additional tables for each category of extracted information.



3.1. Illustration: ER diagram of the enriched trial database

3.1.1 Main table

Trials are indexed with their trial IDs as their primary keys. In addition, there are columns for the trial's phase, status (completed, recruiting, etc...), type (interventional, observational, etc...), their official title, their source (what group, university or corporation that initiated the trial) and the date they were added to clinicaltrials.gov.

3.1.2 Locations

A separate table exists for the trials' locations, i.e. the countries involved in the trials. The reason that a separate table exists for this and that it's not included in the main table is that there are potentially several countries referencing one trial (one to many relationships).

3.1.3 Drugs

Information about extracted drugs have a foreign key referencing their trial's ID, a column containing the name of the drug and a phrase representing the original pre noise filtering context that the drug was extracted from (more details on the implementation section). Lastly, it contains a column indicating whether there is a match for the drug entry in the DGIdb (Drug Gene Interaction database) [28], which is used for looking up search results in the curator tool.

3.1.4 MeSH terms

In addition to the trial ID foreign key, the table representing MeSH terms consists of a column for the MeSH term itself, as well as a column indicating its source (whether it was originally extracted from the trial or if it is a parent of a extracted term – more details in the implementation section).

3.1.5 Abbreviations

This table is rather self-explanatory, containing a column for the abbreviation and the phrase it abbreviates. This table was included to assist in filtering out noise/false positives when identifying gene symbols.

3.1.6 Genes

Genes are represented with entreZ IDs[29], gene symbols and gene names - both official and alternative. As a result, the gene table has the largest number of columns in the database. In addition to columns representing these fields, a gene entry also contains columns for the original match phrase, for indicating whether the entry has been normalized as well as possible comments and validation markers from the CT curator(chapter 4).

3.1.7 Mutations

The mutation table simply contains a single column containing the mutation phrase in addition to the foreign key.

3.1.8 Additional tables

In addition to these tables, there are tables used for known general drug names, specifically drugs listed in the DGIdb database. This information is used by the CT curator to list potential genes in the same trial that are known to have interactions with the given drug. Additionally, there is a table listing possible relations between genes and mutations being mentioned in the same trials. Lastly, another table is used to keep track of which trials have already been processed when updating the database with new trials

from clinicaltrials.gov (see 3.2.7 on maintenance). These tables are not included in the ER diagram. Tables for the evaluated but not implemented gene indexing methods are not included either.

3.2 Implementation

This sub-chapter will cover the process of how the database was put together. Various problems encountered during implementation and decisions made will also be covered here.

3.2.1 Trial files / main table

ClinicalTrials.gov provides a crawler service that allows users to systematically download every submitted trial. [30] Using the LWP::Simple Perl library [31], a script iterates through the trials and downloads them.

Trials are published in two formats; an XML format and an HTML format. Primarily we will be working with the XML files, but as certain parts are not formatted suitably, the HTML files are also downloaded.

As each trial is successfully downloaded, its trial ID is inserted into the main table. Another script later processes the trials and extracts the basic information (sources, official title etc.) to complete the table.

3.2.2 Countries / locations

Separate tables were created to store the involved countries in each trial as well as the exact given address. Here is a summary of the most involved countries:

3.2.2 Countries / locations

Occurrences	Country
74779	United States
12226	Canada
11299	Germany
10119	France
8687	United Kingdom
6620	Italy
6078	Spain
5283	Netherlands
5091	Belgium
4920	Republic of Korea

There is a clear dominance of US-involved clinical trials, although this is partly attributable to ClinicalTrials.gov being an American site. The US is followed by Canada which is involved in merely 1/6 of the number of trials. After Canada, mostly European countries make the top 10.

More specifically, we can look at the exact address of the involved parties. This gives us an overview of the participation of the individual parties, not just countries in general. This list topped by a good margin by 77030, Houston, Texas, USA, which is the address of The University of Texas Health Science Center at Houston. 5261 trials registers involvement with this address.

Occurrences	Address
5261	77030, Houston, Texas, USA (The University of Texas Health Science Center at Houston)
3297	20892, Bethesda, Maryland, United States (National Institutes of Health Clinical Center)
2801	02115, Boston, Massachusetts, United States (Northeastern University)
2668	19104, Philadelphia, Pennsylvania, United States (University of Pennsylvania)

3.2.2 Countries / locations

Occurrences	Address
2375	55905, Rochester, Minnesota, United States (Mayo Clinic in Rochester)
2374	27710, Durham, North Carolina, United States (Duke University Medical Center)
2343	02114, Boston, Massachusetts, United States (Massachusetts General Hospital)
2203	10021, New York, New York, United States (Memorial Sloan-Kettering Cancer Center)
2114	78229, San Antonio, Texas, United States (University of Texas Health Science Center at San Antonio)
2098	10032, New York, New York, United States (Columbia University Medical Center)

3.2.3 Abbreviations

Locating abbreviations was done using a Python script written by Vincent Van Asch [32]. The script is simply run given the path to the trial folder and recursively processes all trials.

3.2.4 Drug extraction and noise reduction

The XML files contain fields specifically for drugs used in the trial. They are found within “<intervention>” tags where sub-tag “<intervention_type>” is listed as a drug. The drug name itself is then found in the trailing sub-tag “<intervention_name>”. Extracting these are straightforward enough and is done separately by a perl script.

However, there doesn't seem to be any rules or standards as to how drug names are submitted. This presents a problem, as the drug entries are in many cases accompanied by dosage size, intended time interval between doses or other (in our case) superfluous information. Drug entries may even contain whole sentences describing the administration of the drug.

3.2.4 Drug extraction and noise reduction

Two or more drugs may also be listed as one entry contrary to being listed in separate tags. These are divided in arbitrary ways, either by natural language (“and”, ”or”, “combined with”, etc...) or symbols (e.g.: “/”, “+”).

trialID	name
NCT01625871	tablets artemether/lumefantrine
NCT01624142	amg 145
NCT01624142	amg 145
NCT01624246	ceftaroline fosamil/avibactam (cxl)
NCT01625286	azd5363 when combined with weekly paclitaxel.
NCT01625286	azd5363 when combined with weekly paclitaxel.
NCT01625286	azd5363when combined with weekly paclitaxel.
NCT01625286	a placebo in combination with weekly paclitaxel.
NCT01624532	normal saline
NCT01625455	aprepitant
NCT01625455	placebo
NCT01625000	mp-214 low dose
NCT01625000	mp-214 middle dose
NCT01625000	mp-214 high dose
NCT01625000	risperidone
NCT01625000	placebo
NCT01625806	ro4602522
NCT01625806	ketoconazole
NCT01624636	placebo
NCT01624636	lfg316
NCT01624636	lfg316

21 rows in set (0.00 sec)

Illustration 3.2: Samples of original drug entries

As shown in Fig. 3.2, there is a lot of noise in the original entries. Additionally, “placebo” is listed as a drug in several trials, and should be excluded.

To deal with this, the entries are run through a Python filtering script that reduces noise and excludes certain entries. This is done using regular expressions, stop lists and Python string editing functions. A full overview of the drug filtering process is outlined here:

3.2.4 Drug extraction and noise reduction

- Drugs have all non-ASCII characters (drugs are at times trailed by ® and/or) removed as they aren't necessary for our project's purpose and they tend to cause problems with the MySQL database.

- The entries are split on possible separator words and symbols to separate any multiple entries.

- A stop-list is used to filter out unnecessary information (including all “placebo” mentions).

- Another stop-list is used to filter out dosage information; all measurement units are removed. Likewise, all numbers leading a measurement unit are removed.

- Entries are looked up against WordNet [33], a lexical database for English. All words that are common (not drug names) are excluded.

trialID	name
NCT01625871	artemether
NCT01625871	lumefantrine
NCT01624142	amg 145
NCT01624142	amg 145
NCT01624246	ceftaroline fosamil
NCT01624246	avibactam cxl
NCT01625286	azd5363 when paclitaxel
NCT01625286	azd5363 when paclitaxel
NCT01625286	azd5363when paclitaxel
NCT01625286	paclitaxel
NCT01624532	saline
NCT01625455	aprepitant
NCT01625000	mp-214
NCT01625000	mp-214
NCT01625000	mp-214
NCT01625000	risperidone
NCT01625806	ro4602522
NCT01625806	ketoconazole
NCT01624636	lfg316
NCT01624636	lfg316

20 rows in set (0.00 sec)

Illustration 3.3: Samples of filtered drug entries

3.2.4 Drug extraction and noise reduction

Fig. 3.3 shows the samples from 3.2 after being run through an early version of the filtering script. The rows have been affected in several ways. For instance, the multiple-drug containing entry from NCT01625871 has been divided into two separate ones, and the dosage information (“tablets”) has been filtered out. The mp-214 entries are also stripped of dosage information. Both placebo entries have been left out entirely. However, the NCT01625286 entries still contain noise as there is residue from the dosage information that has not been completely filtered out.

After all entries have been filtered, 131133 out of 150804 trials contained one or more drug entries[10]. These are the most common entries:

Drug name	Total occurrences
Cyclophosphamide	1639
Cisplatin	1478
Paclitaxel	1284
Carboplatin	1243
Gemcitabine	1218
Docetaxel	1134
Dexamethasone	952
Bevacizumab	840
Doxorubicin	811
Etoposide	792

3.2.5 MeSH terms extraction and the MeSH tree

MeSH (Medical Subject Header) is a vocabulary used for indexing MEDLINE articles. The vocabulary is constructed as a tree with 16 top level categories, who represent very general subjects, e.g. diseases, drugs/chemicals, organisms, etc. Each category can be divided into several, more specific nodes. These are then divided into other, even more specific subjects, down to as much as twelve levels.

10 As of October, 2013

3.2.5 MeSH terms extraction and the MeSH tree

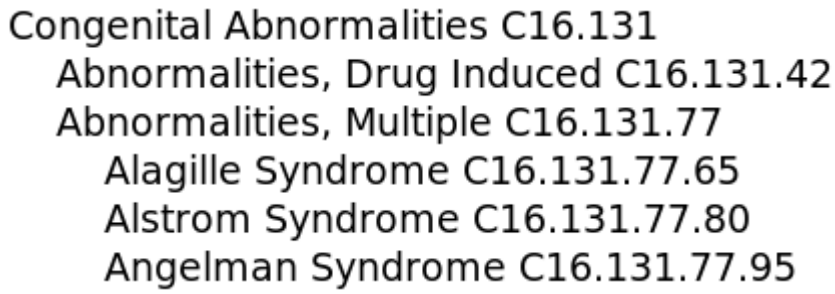


Illustration 3.4: An example of the MeSH tree using Congenital Abnormalities as root node

As in drug extraction, MeSH terms are easily extracted as they are located within pre-defined tags. Clinical trials from clinicaltrials.gov are already indexed with MeSH terms. However, contrary to the HTML files, the XML files do not represent the full set of the MeSH terms, only the leaf nodes. To include the rest of the branch, we look up the MeSH tree and recursively find the terms' parent nodes. A MeSH term may also be a child of several parent terms. For example, 'Leukemia, Lymphoid' is a child of 'Leukemia' as well as two different nodes representing 'Lymphoprofilerative Disorders' (also ambiguous entries), where one of these is a child of 'Lymphatic Diseases' and the other is a child of 'Immunoproliferative Disorders'. This shows that a single MeSH term can branch up to several subcategories and root nodes. When dealing with these ambiguities, all parent branches are included.

Here is a summary of the most common MeSH terms, both entries extracted directly from the XML files and entries from the extended MeSH tree:

MeSH_term	XML	Extended	Total
Neoplasms	5881	33482	39363
Pathological Conditions, Signs and Symptoms	0	33825	33825
Organic Chemicals	0	26595	26595
Cardiovascular Diseases	1948	18920	20868

3.2.5 MeSH terms extraction and the MeSH tree

MeSH_term	XML	Extended	Total
Heterocyclic Compounds	0	20234	20234
Nervous System Diseases	0	18020	18202
Immune System Diseases	45	17909	17954
Neoplasms by Histologic Type	7	17327	17334
Pathologic Processes	8	17325	17333

3.2.6 Genes

For annotating genes in the trials, several different approaches were tested and evaluated. A basic dictionary-based approach was programmed that matches text against the NCBI dictionary of human genes. Additionally, three third-party NER systems were tested; GNAT, BeCAS and Gimli.

To evaluate the various approaches, a set consisting of 100 trials was selected, where each trial had one or more genes found by the initial dictionary-based approach. The trials were then manually curated, and the results were compared against the automatic annotation.

The dictionary-based approach works by simply loading the 46315 different genes into a Perl hash, including all alternative symbols, descriptions and alternative descriptions. The script then matches every sentence for partial or complete matches of gene names, and each word for matches against symbols. Matches are then normalized against the gene ID, main symbol and main description of the gene. The initial idea of using POS tagging to avoid redundant lookups (i.e. only look up nouns and noun phrases – see 2.1.2) was discarded as the potential time saved in runtime was not worth the

3.2.6 Genes

development time that would go into finding and implementing a suitable (or programming a new, domain-specific) POS tagger. As this approach does not consider context, a very low precision level was achieved.

Recall	84.47%
Precision	54.65%
F-score	66.37%

GNAT was dependent on a remote web service which consistently seemed to go offline. Even when results were produced, they were erratic. Albeit these poor results were most likely attributable to poor configuration on my part, after consistent failure with GNAT, the idea of using the system was discarded in order to put time into trying other tools. As such, no test data was produced for GNAT.

Gimli NER requires pre-processing before doing the actual annotation. First, the text has to be formatted into one sentence per line. Gimli then converts this file into a so-called corpus. This is done using the Genia Dependency Parser (gdep). Formatted corpora are formatted with a unique identifier separated with a whitespace from the sentence, like this (for each sentence):

"NCT00326339 R406 is metabolized by CYP3A4, and ketoconazole increases the R406 AUC of a dose of R788 by approximately 2 fold."

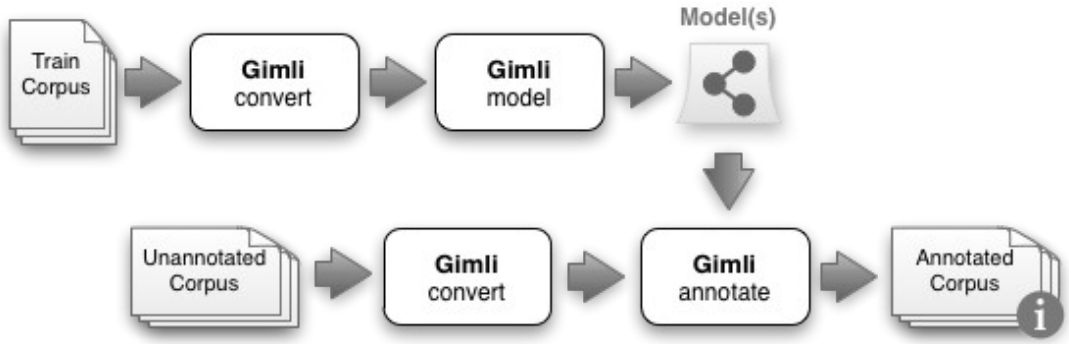


Illustration 3.5: Gimli workflow, from bioinformatics.ua.pt/support/gimli/doc/

3.2.6 Genes

Gimli then uses one or several models to annotate the corpus. The system can train models, and there are trained models available. In our case, we are using a model trained on BC2, using forward parsing. When Gimli is done annotating, the output of our example sentence will look like this:

“NCT00326339|18 24|CYP3A4”

where 18 and 24 are the positions of the first and last char of the entity. For normalization, the match phrases were mapped against the gene symbols/descriptions using a part of the code for the basic dictionary-based approach.

Gimli reports an F-score of 87,54% on BioCreative test data and 73,05% on JNLPBA test data. (This gap between BC2 and JNLPBA performance is typical. [34]) On clinical trials, this is how Gimli performed:

Recall	83.71%
Precision	75.42%
F-score	79.35%

Clearly Gimli has an advantage over the basic symbol matching method. While the F-Score does not compete with the results from the BC2 test sets, it exceeds the performance on JNLPBA.

BeCAS is easily implemented through a python package or a script, and in contrast to Gimli does not require the data to be converted prior to annotation or pre-trained models. Rather, the data is submitted to a web API and processed remotely. For normalization, BeCAS returns a UniProt ID along with the annotated concept. This UniProt can be converted to a Gene ID, which can then be used to retrieve the gene's symbols and descriptions. However, using this method to normalize gave worse results than using the same method that was used on the Gimli results. This is how BeCAS performed on the test data:

Recall	77.65%
Precision	71.43%
F-score	74.41%

Naturally, as Gimli excels both in recall and precision, it was chosen for use in the automated annotation.

3.2.7 Mutations

To extract mutations from the trials, a rule-based system named MutationFinder [35] was applied. It is available in Python, Perl and Java. We have included the Python version in our pipeline as it is easily downloaded and installed as a separate module that can be addressed programmatically.

MutationFinder only extracts point mutations. Extending the approach to other types of mutations would introduce a new problem; as point mutations are named based on rules (see 2.2.1), they are easily detected in free text. However, other types of mutations are more ambiguous and there aren't many readily available open-source packages dealing with them.

The point mutations detected by MutationFinder are stored in the database. Another script then attempts to couple the mutations to genes that are also detected in the same trials. This is done by looking up the sequence of amino acids of the gene candidates. A potential relation between a gene an the mutation is then easily detected by checking the given amino acid at the given position (codon) in the mutation name. If the amino acid in the gene sequence is the same as the original amino acid in the mutation name, the mutation may be corresponding to the gene. Let's look at an example; we have an occurrence of the mutation V600E and the BRAF gene in the same trial[11]. Here is an outline of the process for determining gene-mutation relationship: First we look up the amino acid sequence for the BRAF gene:

11 E.g.: NCT01827384

3.2.7 Mutations

MAALSGGGGGGAEPGQALFNGDMEPEAGAGAGAAASSAADPAIPEEVWNIKQMIKLTQEHIEALLDKFGGEHNPPSIYLEAYEEYTSKLDALQQRE-
QQLLESLGNGTDFSVSSSASMDTVTSSSSSSLSVLPSSLSVFQNPTDVARSNPKSPQKPIVRVFLPNKQRTVVPARCGVTVRDSLKKALMMR-
GLIPECCAVYRIQDGEKKPIGWDTDISWLTGEELHVEVLENVPLTTHNFVRKTFFTLAFCDFCRKLLFQGFRCQTCGYKFHQRCSTEVPLMCVNY-
DQLDLLFVSKFFEHHPIPQEEASLAETALTSGSSPSAPASDSIGPQILTSPSPSKSIPIPQPFRPADEDHRNQFGQRDRSSSAPNVHINTIEPVNID-
DLIRDQGFRGDGGSTTGLSATPPASLPGSLTNVKALQKSPGPQRERKSSSSSEDRNRMKTLGRRDSSDDWEIPDGQITVGQRIGSGSFGTVYKGK-
WHGDVAVKMLNVTAPTPQQLQAFKNEVGVLRKTRHVNILLFMGYSTKPQLAIVTQWCEGSSLYHHLHIIETKFEMIKLIDIARQTAQGMDYLHAKSI-
IHRDLKSNNIFLHEDLTVKIGDFGLATVKSRWSGSHQFEQLSGSILWMAPEVIRMQDKNPYSFQSDVYAFGIVLYELMTGQLPYSNINNRDQIIFMVGR-
GYLSPDLSKVRSNCPKAMKRLMAECLKKKRDERPLFPQILASIELLARSLPKIHRSASEPSLNRAGFQTEDFSLYACASPKTPIQAGGYGAFPVH

We then locate the amino acid at the position given in the mutation's name: the 600th amino acid is valine, which corresponds with the mutation. Thus, the V600E mutation - or every V600* mutation - can potentially be related to the BRAF gene. Information representing this relationship is stored in a junction table between the gene table and the mutation table.

After all mutations were matched again potential genes, barely half of the mutations matched up against a gene; only 712 out of 1298 mutations had a potential gene candidate in the same trial. This could be caused by poor gene annotation (a lack of recall resulting in the mutation's gene not being detected), and/or poor mutation annotation (a lack of precision resulting in a false positive mutation match).

Total number of mutations	1298
Mutations matched against one or more genes	712
Mutations without matches	586
% of matched mutations	54.9%

A total of 461 different unique mutations were registered, the most prominent being the V600E mutation. Other frequent mutations are summarized here:

Mutation	**Occurrences**
V600E	65
L858R	46
T315I	43

3.2.7 Mutations

Mutation	Occurrences
C282Y	35
M184V	32
T790M	29
K65R	27
G20210A	25
P13K	23
H63D	21

Additionally, we can see from combining the gene and mutation tables that the majority of V600E annotations (77%) are found in trials where a mention of the BRAF gene is also found. The BRAF gene is found in a total of 228 unique trials, meaning 22% of BRAF-related trials also include the mutation. This information reflects the fact that the V600E mutation in the BRAF gene is a common mutation known to cause certain types of cancer and is being researched a lot.

Occurrences	Gene
50	BRAF
6	ALB
1	PMEL

Among other genes that can contain a V600E mutation, albumin also occasionally appear in the same trials as the mutation.

3.2.8 Maintenance

Naturally, an important part is to keep the database up-to-date with newly published clinical trials. In order to ensure that the most recent clinical trials are available and annotated, a script will download every new trial, process them and store the results. The curator tool is then automatically updated to include these trials.

3.2.8 Maintenance

As the results from certain scripts are dependent on the results of other scripts, the maintenance makes sure the scripts are run in the correct order. They are run in the following order:

1. The XML files are downloaded
2. Basic information is extracted from the trials (title, phase, source, status, date and study type (observational vs interventional)).
3. Raw/free text is extracted from the XML files, i.e. the brief summary, detailed description and eligibility criteria sections.
4. Countries and locations/addresses are extracted
5. Drugs are extracted from the trials
6. Drug entries are filtered and matched against the DGIdb table
7. MeSH entries are extracted
8. MeSH trials are looked up against the MeSH tree, extracting parent terms and adding them to the database
9. Mutations are extracted using MutationFinder
10. Abbreviations are extracted and loaded
11. NER packages for gene mention detection are run
12. Gene entries are normalized, i.e. entries are matched against the NCBI gene name list and associated with a gene symbol
13. Potential links between detected mutations and genes are found

This script should be run at least on a weekly basis to ensure the database is current. Typical running time if run weekly is roughly 20-40 minutes.

3.2.8 Maintenance

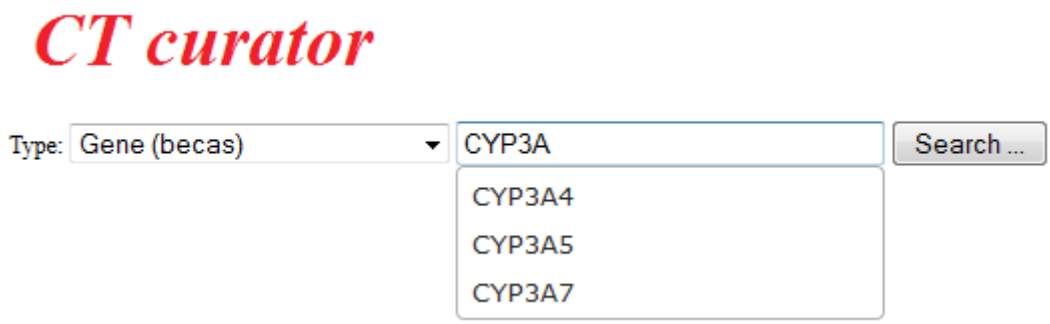
Chapter 4 CT curator

This chapter will cover the functions of the CT curator web tool and how it is implemented. The CT curator is available on http://invitro.titan.uio.no/clinicaltrials

4.1 Posting queries

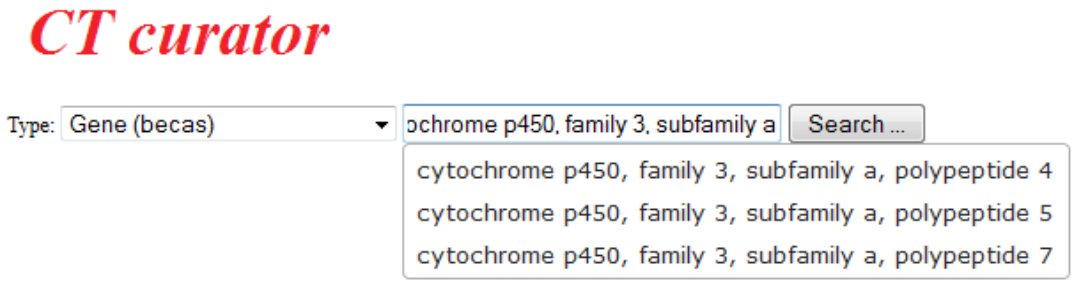
The CT curator provides a user interface for looking up the information extracted from the trials. It is done simply by choosing a class of items from a drop-down list (gene/protein, mutation, drug or MeSH term) and then typing the search term.

A search suggestion feature will recommend entries from the database if two or more characters are entered. For example, by selecting genes as our class and entering the phrase “CYP3A”, a list of the genes in the CYP3A cluster will appear as long as they have been found in one or more trials.



Similarly, typing “cytochrome p450, family 3, subfamily a” will return the same list, only represented as gene descriptions instead of symbols. Terms can either be selected from this list or written manually.

4.1 Posting queries



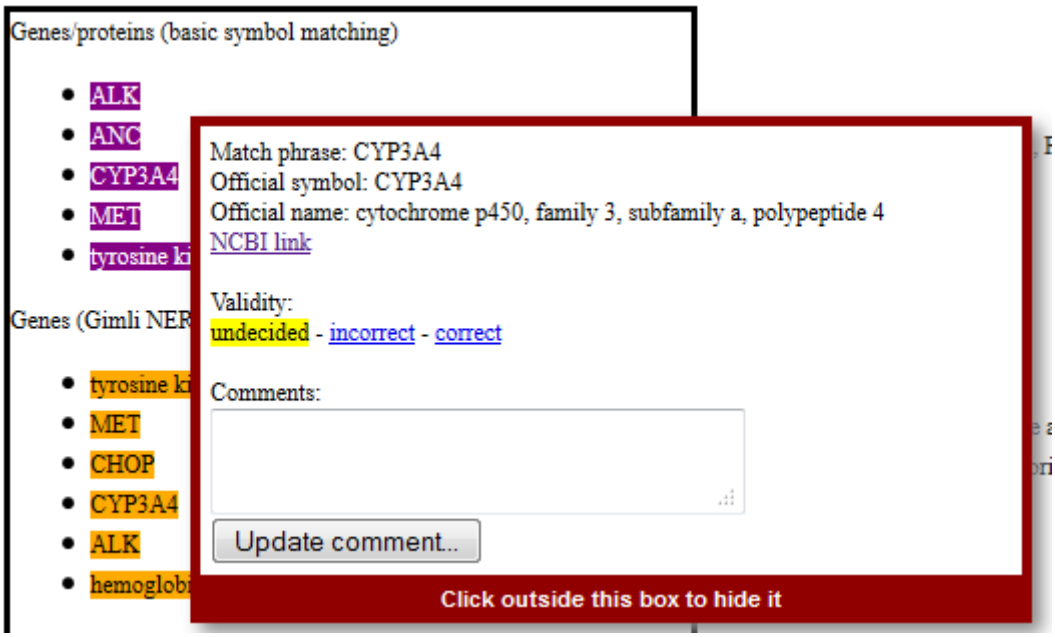
When the query is posted, a list is returned with trials containing the search term. These trials are sorted chronologically and a superficial representation is given, containing trial ID, phase, date received by clinicaltrials.gov, their official title as well as a flag for each nation involved. Results can then be filtered by country and phase simply by selecting the values from two drop-down lists at the top of the results.

Following the “more details...” link will open a more detailed representation of the selected trial. This view also includes the exact locations of the involved parties, brief summary, detailed description, eligibility criteria section as well as a list of annotations.

4.2 Description of each group

Each class of annotations (genes, drugs, mutations and MeSH terms) are represented in a sidebar as lists of items. Genes and mutations are also highlighted in the free text. Some classes includes a tool tip with more details. For example, when highlighting genes, a tool tip will appear including the following details:

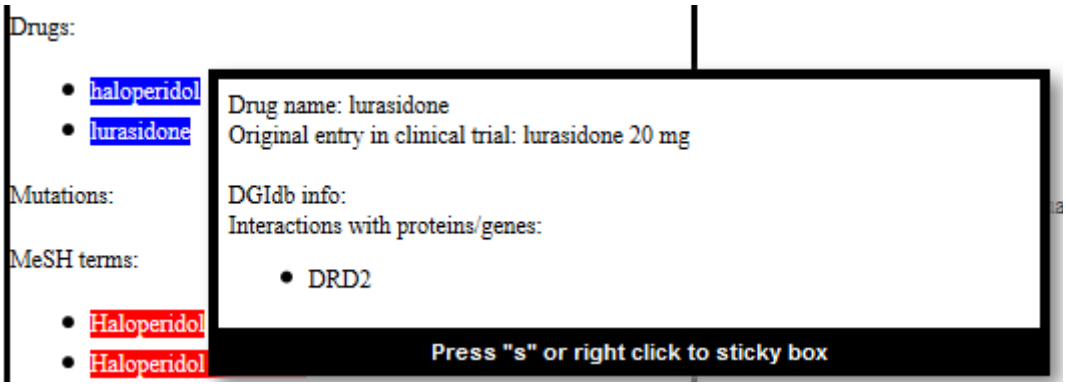
- The original match phrase from the free text
- The official symbol (if found)
- The official gene description (if found)
- A link to the NCBI page on this gene



Drugs are marked in blue. Highlighting them will display a tool tip including the following details:

- The original phrase found in the clinical trial XML file

- The drug entry after being filtered for superfluous information (see 3.2.3). These entries are used when looking up search terms.
- Information from the DGIdb about known interactions with protein/genes



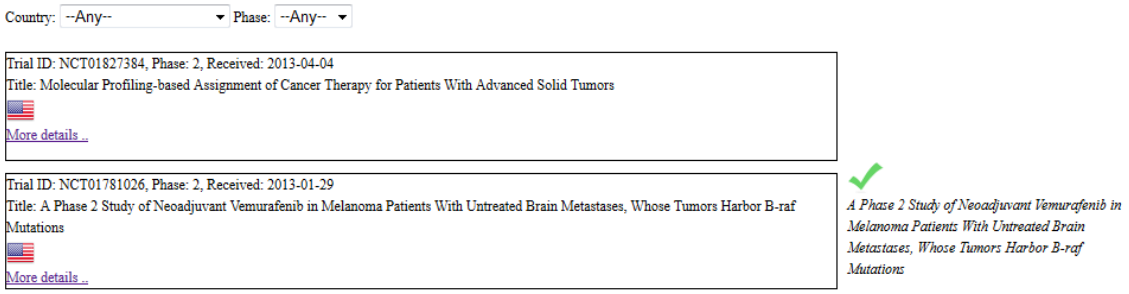
Mutations include the information obtained from the process in 3.2.6. If any potential genes appear in the same trial, these are listed in the mutation's tool tip.

MeSH terms are listed in bright red and dark red; a bright red background indicates that the item was found in the original XML file. A dark red background indicates that this item is a parent MeSH term extracted from one of the original terms (see 3.2.4).

4.3 Interacting with trials and annotations

Comments can be added by users through the ct curator, either to trials as a whole, or individual annotations. While viewing the detailed version of a trial, comments can be added simply by typing the comments in the “user comments” box beneath the trial title / summary. Comments will then be visible for other users, both in the detailed view and when the trial is a part of a set of returned search results.

4.3 Interacting with trials and annotations



Additionally, gene annotations can be marked as correct or incorrect (“undecided” by default). Annotations that are false can also be removed completely. This can be done by opening a gene tool tip and clicking the corresponding link. Trials as a whole can also be marked curatable by marking the check box at the top of the view. A check mark will then appear next to the trial when it is part of a set of returned search results. Lastly, users can add gene/protein and drug annotations that are overlooked by the system. MeSH terms are not editable for the simple reason that they are fetched from an unambiguous list which leaves no possibility for false positives.

4.3 Interacting with trials and annotations

Chapter 5 Discussion

Using various text mining tools, I have developed a system for assisted curation of clinical trials. These annotations are accessible through a web interface where they can be edited or deleted. Creating manual annotations is also possible.

There is clearly room for improvement in the automatic annotation. While there are no concrete numbers available to measure the performance of the annotation of mutations (i.e. F-score), by looking at the number of found mutations that have no corresponding genes in the same trial, one can assume that there is a lack of precision. It would be interesting to evaluate additional pattern-based approaches for detecting mutations.

While the performance of the annotation of genes was acceptable, it would be interesting to see if Gimli could be tweaked to improve precision and recall, e.g. by testing different models in addition to the default BC2 trained model. The algorithm for normalizing genes could also be improved.

In retrospect, more specific methods for selecting the evaluation set may have been more appropriate, e.g. focusing on cancer-related trials using MeSH terms (“Neoplasms”, etc.) as a filter. Additionally, the dictionary-approach was riddled with false positives, resulting in trials with no actual genes being included in the set.

The CT curator itself could also benefit from some improvements. While it provides an interface for editing existing annotations, it does not support definition of associations/relations between bio-concepts. Particularly pharmacogenomic relationships would be useful. Also, while it is functional, it is very primitive and written in hand coded PHP and JavaScript. As was suggested, using a web framework could make the CT curator more user friendly and efficient.

1: A. D. Corlan Medline trend: automated yearly statistics of PubMed results for any query, 2004, http://dan.corlan.net/medline-trend.html
2: B. Alex, C. Grover, B. Haddow, M. Kabadjov, E. Klein, M. Matthews, S. Roebuck, R. Tobin, X. Wang : Assisted curation: Does text mining really help? - 2008
3: Jiao Li, Zhiyong Lu : Systematic identification of pharmacogenomics information from clinical trials - 2012
4: Lawrence M. Friedman, Fundamentals of Clinical Trials, 4th Edition, 2010
5: clinicaltrials.gov, http://clinicaltrials.gov
6: Ioannis Korkontzelos, Tingting Mu, Angelo Restificar, Sophia Ananiadou : Text Mining for Efficient Search and Assisted Cretion of Clinical Trials - 2011
7: I. Kortontzelos, T. Mu, S. Anoniadou : ASCOT: a text mining-based web-service for efficient search and assisted creation of clinical trials - 2011
8: Xiaohong Cao, Karen B Maloney, Vladimir Brusic : Data mining of cancer vaccine trials: a bird's-eye view - 2008
9: Asba Tasneem, Laura Aberle, Hari Ananth, Swati Chakraborty, Karen Chiswell, Brian J. McCourt, Ricardo Pietrobon : The Database for Aggregate Analysis of ClinicalTrials.gov (AACT) and Subsequent Regrouping by Clinical Specialty - 2012
10: ctdbase.org entry for Parkinson's Disease, http://ctdbase.org/detail.go?type=disease&acc=168600
11: C. N. Arighi, P. M. Roberts, S. Agarwal, S. Bhattacharya, G. Cesareni, A. Chatr-aryamontri, S. Clematide, P. Gaudet, M. G. Giglio, I. Harrow, et al. : An overview of the BioCreative 2012 Workshop Track III: interactive text mining task - 2013
12: Salgado D, et al : MyMiner: a web application for computer-assisted biocuration and text annotation - 2012
13: Settles, B. : ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text - 2005
14: Gerner, M. et al : LINNAEUS: a species name identification system for biomedical literature - 2010
15: C. Jonquet, N. H. Shash, M. A. Musen : The Open Biomedical Annotator - 2009

16: Steven Bird, Ewan Klein, Edward Loper, Natural Language Processing with Python --- Analyzing Text with the Natural Language Toolkit, 2009
17: Daniel Jurafsky, James H Martin, Speech and language processing, 2008
18: Y. Garten, A. Coulet, R. Altman : Recent progress in automatically extracting information from the pharmacogenomic literature - 2010
19: NCBI gene list, http://www.ncbi.nlm.nih.gov/gene
20: A. Taylor, M. Marcus, B. Santorini, The Penn Treebank: An Overview, 2003
21: U. Leser, J. Hakenberg : What makes a gene name? Named entity recognition on the biomedical literature - 2005
22: K. Glette, J. Torresen, T. Gruber, B. Sick, P. Kaufmann, M. Platzner : Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control - 2008
23: Miwa M, Pyysalo S, Ohta S, Ananiadou S : Wide coverage biomedical event extraction using multiple partially overlapping corpora - 2013
24: G. Gonzalez, R. Leaman : BANNER: An executable survey of advances in biomedical named entity recognition - 2008
25: J. Hakenberg et al : The GNAT library for local and remote gene mention normalization - 2011
26: D. Campos, S. Matos, J. Oliveira : Gimli: open source and high-performance biomedical name recognition - 2013
27: T. Nunes, D. Campos, S. Matos, J. Olivera : BeCAS: biomedical concept recognition services and visualization - 2013
28: DGIdb homepage, http://dgidb.genome.wustl.edu/
29: D. Maglott, J. Ostell, K.D. Pruitt, T. Tatusova : Entrez Gene: gene-centered information at NCBI - 2004
30: Clinicaltrials.gov crawler service, http://www.clinicaltrials.gov/ct2/crawl
31: LWP::Simple Perl library, http://search.cpan.org/~gaas/libwww-perl-6.05/lib/LWP/Simple.pm
32: Vincent Van Asch - scripts, http://www.clips.ua.ac.be/~vincent/software.html
33: C. Feillbaum, WordNet and wordnets, 2005

34: S. Dingare, M. Nissim, J. Finkel, et al. : A system for identifying named entities in biomedical text: How results form two evaluations reflect both the system and the evaluation - 2004
35: J. Gragory Caporaso, W. A. Baumgartner, David A. Randolph, K. Bretonnel Cohen, L. Hunter : MutationFinder: a high-performance system for extracting point mutation mentions from text - 2007